REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
25-11-2014	Final Report	1-May-2010 - 30-Apr-2014

4. TITLE AND SUBTITLE
Final Report: Research Areas 5: Securing Untrusted Binaries with Acceptance Testing and Field Monitoring

5a. CONTRACT NUMBER
W911NF-10-1-0131

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS
Jack W. Davidson, Jason D. Hiser

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
University of Virginia
P. O. Box 400195

Charlottesville, VA 22904 -4195

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
56439-CS.9

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
Today's Army relies on computing to effectively engage an increasingly sophisticated enemy. Using commercial off the shelf (COTS) software to build Army systems has many advantages: reduced development costs, leveraging of vendor resources and expertise, and greater functionality. Unfortunately, COTS software often includes untrusted components that may contain any variety of uncaught coding errors, intentionally planted time or logic bombs, trojan horses, backdoors, or other features which can cause security violations.

15. SUBJECT TERMS
Final Report, Acceptance Testing, Field Certification, Cyber security

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Jack Davidson
UU	UU	UU	UU		19b. TELEPHONE NUMBER 434-982-2209

Report Title

Final Report: Research Areas 5: Securing Untrusted Binaries with Acceptance Testing and Field Monitoring

ABSTRACT

Today's Army relies on computing to effectively engage an increasingly sophisticated enemy. Using commercial off the shelf (COTS) software to build Army systems has many advantages: reduced development costs, leveraging of vendor resources and expertise, and greater functionality. Unfortunately, COTS software often includes untrusted components that may contain any variety of uncaught coding errors, intentionally planted time or logic bombs, trojan horses, backdoors, or other features which can cause security violations.

In this research, our goal is to provide high levels of software assurance for mission-critical software systems through a novel concept called "field certification" of software. Field certification is a particular instance of a powerful, general approach that we have developed for providing software assurance. Our general approach for providing high levels of software assurance is to logically interpose a small, trusted software component between the application and the operating system and use this component to enforce
specific program properties to ensure the proper operation of the software application and prevent vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software from being exercised intentionally (by an malicious adversary) or unintentionally (by a non-malicious user).

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in peer-reviewed journals:

(b) Papers published in non-peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in non peer-reviewed journals:

(c) Presentations

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received	Paper

TOTAL:

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received	Paper
08/31/2012	1.00 Anh Nguyen-Tuong, Michele Co, Matthew Hall, Jason Hiser, Jack W. Davidson. ILR: Where'd My Gadgets Go?, 2012 IEEE Symposium on Security and Privacy (SP) Conference dates subject to change. 20-MAY-12, San Francisco, CA, USA. : ,
09/19/2013	3.00 Benjamin D. Rodes, Anh Nguyen-Tuong, Jason D. Hiser, John C. Knight, Michele Co, Jack W. Davidson. Defense against Stack-Based Attacks Using Speculative Stack Layout Transformation, Run-time Verfication 2012. 25-SEP-12, . : ,
09/19/2013	2.00 Sudeep Ghosh, Jason Hiser, Jack W. Davidson. Software protection for dynamically-generated code, the 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop. 26-JAN-13, Rome, Italy. : ,
11/25/2014	6.00 R Rajkumar, A Wang, J D Hiser, Anh Nguyen-Tuong, J W Davidson, J C Knight. Component-Oriented Monitoring of Binaries for Security, 2011 44th Hawaii International Conference on System Sciences (HICSS 2011). 04-JAN-11, Kauai, HI. : ,
11/25/2014	4.00 Anh Nguyen-Tuong, Jason D. Hiser, Michele Co, Nathan Kennedy, David Melski, William Ella, David Hyde, Jack W. Davidson, John C. Knight. To B or not to B: Blessing OS Commands with Software DNA Shotgun Sequencing, 2014 Tenth European Dependable Computing Conference (EDCC). 13-MAY-14, Newcastle, United Kingdom. : ,
11/25/2014	7.00 Sudeep Ghosh, Jason D. Hiser, Jack W. Davidson. What's the PointiSA?, Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security. 11-JUN-14, Salzburg, Austria. : ,
11/25/2014	8.00 Anh Nguyen-Tuong, Michele Co, Benjamin Rodes, Matthew Hall, Clark L. Coleman, John C. Knight, Jack W. Davidson, Jason D. Hiser. A Framework for Creating Binary Rewriting Tools (Short Paper), 2014 Tenth European Dependable Computing Conference (EDCC). 13-MAY-14, Newcastle, United Kingdom. : ,
TOTAL:	**7**

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

Received Paper

TOTAL:

Number of Manuscripts:

Books

Received Book

TOTAL:

Received Book Chapter

TOTAL:

Patents Submitted

Methods, systems and computer readable media for detecting command injection attacks

System, Method & Computer Readable Medium for Software Protection via Composable Process-level Virtual Machines

Patents Awarded

Awards

Best paper award for "Software Protection for Dynamically-generated Code at 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop, Rome,Italy, January 2013.

Graduate Students

NAME	PERCENT_SUPPORTED	Discipline
Matthew Hall	1.00	
FTE Equivalent:	**1.00**	
Total Number:	**1**	

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Jack W. Davidson	0.08	No
Jason D. Hiser	0.05	
John C. Knight	0.02	
FTE Equivalent:	**0.15**	
Total Number:	**3**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period: 0.00

The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:..... 0.00

The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:..... 0.00

Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):..... 0.00

Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:..... 0.00

The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense 0.00

The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:..... 0.00

Names of Personnel receiving masters degrees

NAME
Matthew Hall
Total Number: 1

Names of personnel receiving PHDs

NAME

Total Number:

Names of other research staff

NAME PERCENT_SUPPORTED

FTE Equivalent:
Total Number:

Sub Contractors (DD882)

Inventions (DD882)

5 **Methods, systems and computer readable media for detecting command injection attacks**

Patent Filed in US? (5d-1) Y

Patent Filed in Foreign Countries? (5d-2) N

Was the assignment forwarded to the contracting officer? (5e) N

Foreign Countries of application (5g-2):

5a: Anh Nguyen-Tuong

5f-1a: University of Virginia

5f-c: 85 Engineer's Way

Charlottesville VA 22904

5a: Jason D. Hiser

5f-1a: University of Virginia

5f-c: 85 Engineer's Way

Charlottesville VA 22904

5a: Michele Co

5f-1a: University of Virginia

5f-c: 85 Engineer's Way

Charlottesville VA 22904

5a: John C. Knight

5f-1a: University of Virginia

5f-c: 85 Engineer's Way

Charlottesville VA 22904

5a: Jack W. Davidson

5f-1a: University of Virginia

5f-c: 85 Engineer's Way

Charlottesville VA 22904

5 **System, Method & Computer Readable Medium for Software Protection via Composable Process-level Virtual Machines**

Patent Filed in US? (5d-1) Y

Patent Filed in Foreign Countries? (5d-2) N

Was the assignment forwarded to the contracting officer? (5e) Y

Foreign Countries of application (5g-2):

5a: Jack W. Davidson

5f-1a: University of Virginia

5f-c: 85 Engineer's Way

Charlottesville VA 22904

5a: Jason D. Hiser
5f-1a: University of Virginia
5f-c: 85 Engineer's Way
Charlottesville VA 22904

5a: Sudeep Ghosh
5f-1a: University of Virginia
5f-c: 85 Engineer's Way
Charlottesville VA 22904

Scientific Progress

See Attachment

Technology Transfer

See Attachment

Research Area 5: Securing Untrusted Binaries with Acceptance Testing and Field Monitor
Proposal Number 56439-CS
Jack W. Davidson, University of Virginia

Objective

A major problem for the Army is the inability of software testing to produce full trust in a software system, and the resulting need to deploy software that is untrusted and thus subject to malicious attack and failure. This research has aimed to increase the trustworthiness of software at deployment, while providing an environment in which the (still not fully trusted) software can operate safely after deployment despite the presence of malware or exploitable vulnerabilities.

In particular, this research has:

- Developed methods to ensure that portions of an untrusted program that have not undergone thorough acceptance testing (which could include time bombs, logic bombs, back doors, viruses, etc.) cannot be executed in an insecure manner,
- Allowed installation and distribution of untrusted software such that use of the software helps define normal behavior of the program, without compromising system security, and
- Provided mechanisms that allow the system administrator to learn about the safe execution of the programs and authorize additional program features post-deployment.

Approach

Our general approach for providing high levels of software assurance is to logically interpose a small, trusted software component between the application and the operating system (see Figure 1) and use this component to enforce specific program properties to ensure the proper operation of the software application and prevent vulnerabilities, either intentionally or unintentionally designed or inserted as part of the software, from being exercised intentionally (by a malicious adversary) or unintentionally (by a nonmalicious user). The trusted software component, called Strata, is a highly efficient software dynamic translator that provides facilities for efficiently monitoring and dynamically modifying a program as it executes.

The key idea is to use the acceptance-testing phase of the software lifecycle to determine the program properties that characterize the normal behavior (i.e., non-malicious) behavior of the application. After acceptance testing, the application, a characterization of its acceptable behavior, and a Strata-based Field Certification Software Dynamic Translator (FC-SDT) is deployed. FC-SDT ensures that the application can only perform operations that were determined or specified during the acceptance-testing phase. The following sections provides more details about the various components and the research challenges that must be addressed for field certification to be a viable approach for deploying potentially untrusted code.

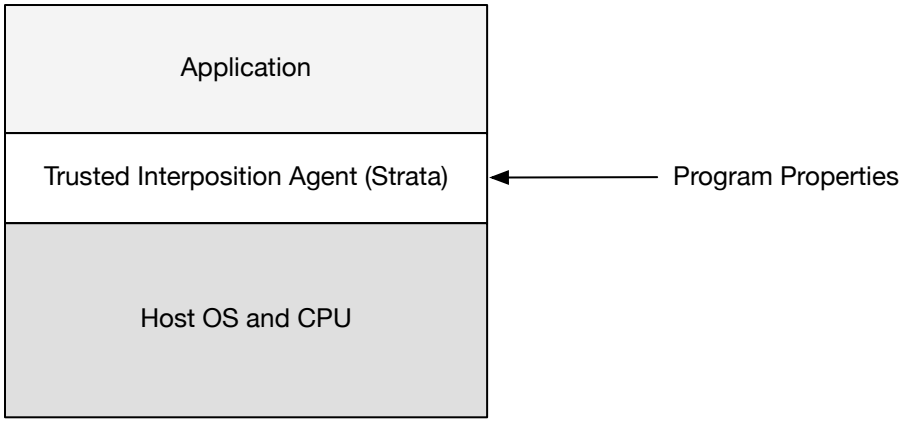

Figure 1: Trusted software layer interposed between the application and the OS.

Scientific Opportunities and Barriers

The performed research has worked to address many research opportunities and challenges, including:

- Determining the run-time properties and behaviors that characterize operation of each of several classes of malware (e.g., Trojan horses, back doors, time bombs, logic bombs, viruses, code injection attacks, arc-injection attacks, etc.),
- Determining the synergy possible between static and dynamic analyses in the identification, observation and enforcement of program properties,
- Developing appropriate criteria and mechanisms for determining whether the pre-deployment testing phases has sufficiently established normal application behavior,
- Determining the run-time properties and behaviors that can be disallowed without generating false positives for non-malicious applications,
- Determining run-time enforcement policies for each suspicious or malicious behaviors that will achieve the goal of thwarting malware while not disrupting normal program use,
- Determining the interaction between higher-level monitoring and lower-level monitoring policies, and
- Developing ways to structure and convey information between the acceptance testing and the field certification components.

Significance

The Army needs high assurance that the programs it uses on sensitive data or for military applications cannot execute malicious code. This requirement has previously limited the use of uncertified programs. Our techniques allow uncertified software on many more systems with a significantly improved level of security. We believe that the concept of using acceptance testing to define the normal behavior of a program is a significant advancement to the state of the art of software assurance. Delaying full certification of the software until

post-deployment provides the unique ability to safely execute an uncertified program in an environment with secure data.

Accomplishments

We have made significant progress during our work on the project. In particular, we have:

- Designed a system for collecting traces of the application to help characterize normal behavior. In each mechanism, we have traces that end at a system call that contain various useful information. In particular, we have focused on a variety of information to hold within the trace:
  - System call execution,
  - Function invocation,
  - Basic block execution, and
  - Call stack history.
- To help assist the acceptance tester, we have used automatic test generation techniques. In particular, we have used "concolic" execution and fuzzing of the program to generate high-coverage acceptance testing.
- We have identified a key issue in determining normal behavior of a program based on recording program traces. This problem is generally label as “path explosion,” relating to the potentially infinite number of possible program executions. We find that even a single input has sources of randomness that result in a large number of paths due to system calls such as `time(), date(), gettid(), getuid()`, etc.
- We used the system to produce a prototype system for handling a special class of uncertified software that is widely used—browser plug-ins. The system, called COMB (Component-Oriented Monitoring of Binaries), provides fine-grained monitoring of these untrusted components. The system is unique in that it uses context information collected as the application runs to determine which components are executing and apply component-specific security policies. A paper describing a preliminary prototype was presented at the Information Security and Cybercrime track at the 44th International Hawaii Conference on System Sciences.[1]
- We have investigated several mechanisms for dealing with the "path explosion problem", a situation that potentially lead to an infinite number of possible paths through a program, and practically results in a very large number of program paths. In particular, we have investigated:
  - Limiting the length of the path,
  - Summarizing paths using regular expressions,
  - Using paths of length 1, which is equivalent to characterizing which basic blocks in a program represent normal behavior, and
  - Combining the above with call stack history.

[1] R. Rajkhumar, A. Wang, J. D. Hiser, A. Nguyen-Tuong, J. W. Davidson, and J. C. Knight. Component-Oriented Monitoring of Binaries for Security. *Proceedings of the 44th Hawaii International Conference on System Sciences*, Kauai, HI, January 2011, pp. 1–10.

- Findings indicate that it very challenging to fully characterize all of a program, even with help from an acceptance tester and automatic test generation techniques. Consequently, we have extended our system design to include mechanisms to automatically and safely allow continued execution of unsafe program portions. We have preliminary results on a variety of continued execution techniques that prevent malicious programs from controlling the system when online detection mechanisms:
    - When branch instruction in the program attempts to jump to unsafe or unauthorized code, we can force the program to transfer control to a tested or known-good target of the branch. We have several example programs where this mechanism yields programs that continue to operate correctly, and other programs where appropriate error-handling messages are reported.
    - When a branch instruction in a program attempts to jump to unsafe or unauthorized code, we can attempt to force the containing function to automatically return an error code. While this technique is still preliminary, we have one example where this may work better than the previous technique.
- We have ported our initial prototypes based on Strata, to use PEASOUP's Strata/SPRI infrastructure. This change has allowed additional analysis and recovery mechanisms to be implemented on programs shared libraries, not just the main program code. In addition, the integration within this infrastructure allows testing or "vetting" of policies before deployment.
- Our system design includes mechanisms to automatically and safely allow continued execution of unsafe program portions. We have further refined our initial approaches with the deeper analysis of the PEASOUP infrastructure to yield more precise mechanisms for continued execution when unsafe code is detected. These include:
    - Detecting the functions with unsafe code, and determining characteristics of the function, such as the return type (integer, string, struct, array, float, etc.)
    - Characterize common, safe return values from functions with unsafe code (i.e., return -1 as an error code or create an empty string.)
    - For functions with complex return types (i.e., a function which returns a pointer to a structure with an array inside), our characterization technique can also tractably analyze the type, and construct appropriate complex return values.
- Our conversion to use PEASOUP's Strata/SPRI infrastructure has allowed us to test and evaluate our approach on larger bodies of code, such as SPEC's CPU2006 benchmark suite. Initial findings are promising:
    - Several benchmarks are characterized completely -- that is, no unsafe code is detected on new inputs.
    - On other benchmarks, continued execution techniques are successful on several benchmarks.
    - While our technique does yet not handle several programs, we have investigated the failures and anticipate that there are only a few causes of error, and simple extensions can resolve these issues.
- Our system design includes mechanisms to automatically and safely allow continued execution of unsafe program portions. We have further refined our initial approaches with the deeper analysis of the PEASOUP infrastructure to yield more precise mechanisms for continued execution when unsafe code is detected. These include:

- After detecting the functions with unsafe code, determining characteristics of the function (such as the return type), and characterizing common return values, we have evaluated the effectiveness of returning a constant value.

  Key findings Returning constant values from functions that return integers is often effective at keeping the program running and useful. However, returning NULL from a function that returns a pointer is often ineffective. Furthermore, returning a pointer to zeroed memory is also ineffective.

- Programs often expect pointers to be return from a function with particular properties. For example, a function that returns a pointer to a list node may expect that the node has a valid pointer to the node's data. We explored a profiling technique to determine the structure of data that is returned from unsafe functions.

  Key finding Structure of the memory pointed to by returned pointers is important.

- We have continued work on converting our techniques to use PEASOUP's Strata/SPRI infrastructure. Testing on larger bodies of code, such as SPEC's CPU2006 benchmark suite have revealed issues and challenges that were unanticipated. Our results have lead us to a variety of findings:
  - Our techniques are now operating correctly on the entire SPEC CPU2006 benchmark suite, indicating we can analyze and protect common coding constructs, compiler idioms, and instruction sequences.

    Key finding Handling the entire instruction set systematically is important.

  - Our techniques still detect that some program blocks are unsafe.

    Key finding It is unlikely that we all code can be proven safe *a priori*. Recovery techniques that protect the program while not executing unsafe code will be necessary.

- We extended the Field Certification Strata/SPRI infrastructure to support 64-bit code. This allows us to test a wider variety of applications, and also increases transition opportunities.
- We published two papers related to the technology developed under the ARO contract. These papers were presented at the 10th European Dependable Computing Conference in May 2014. They are:

  Framework for Creating Binary Rewriting Tools, J. D. Hiser, A. Nguyen-Tuong, M. Co, B. Rodes, M. Hall, C. L. Coleman, J. C. Knight, and J. W. Davidson. This paper describes a language and framework for rewriting arbitrary binaries so that code that checks and enforces security properties can be easily inserted into the binary either statically or dynamically.

To B or not to B: Blessing OS Commands with Software DNA Shotgun Sequencing, A. Nguyen-Tuong, J. D. Hiser, M. Co, J. W. Davidson, J. C. Knight. The paper describes an innovative and powerful approach that prevents malicious adversaries from executing malicious code within a binary. The approach is inspired by DNA shotgun sequencing that efficiently assembles potential fragments of code to determine if command may be malicious or not. A provisional patent has been filed covering the technique (PCT/US2013/070180 entitled "Methods, Systems and Computer Readable Media for Detecting Command Injection Attacks "). We are extending this technique to handle SQL-injection attacks against web applications. These attacks are #1 on Mitre's list of CWE/SANS list of top 25 Most Dangerous Software Errors and #2 on OWASP's list of Top 10 Most Dangerous Software Errors.

Collaborations and Leveraged Funding

We have been able to leverage other funding sources and collaborations in a variety of ways.

- In our Multi-University Research Initiative (MURI) project, called Helix, we have developed several techniques and prototypes that we have leveraged in this project.
    - First, we have the implemented a *shadow stack.* The shadow stack is technology to monitor the program's activation stack and make shadow copies of return addresses on the stack. We used this low-overhead technology for Field Certification to record or verify the current activation record trace at system call points.
    - To deal with arbitrary binaries, the MURI project developed a tool called the *Stratafier*. The Stratafier inserts the Strata library into any binary so that the program will run under control of Strata. We have needed this technology to meet our goal of dealing with arbitrary binaries.
- In our AFRL-funded project, called Kevlar, we have seen significant hardening and extension of the base technology used for Field Certification:
    - We have improved Strata and the Stratafier to be near-production quality, ready for large-scale deployment. The technology now can safely deal with almost any program, including programs that 1) use signal handling (including nested signal handing), 2) throw exceptions, 3) demand high performance, 4) use threading, 5) use interprocess communication, 6) use the network, and many other program idioms.
    - Further extended the SPRI interface to include a callback mechanism, useful for monitoring function or system calls for potentially malicious parameters.
    - Part of the Kevlar project is to port current technologies to a Windows platform, further extending reach of the base technology for Field Certification. Windows support will greatly enhance the impact of Field Certification.

- Another project that we are working on is called Preventing Exploits Against Software of Unknown Provenance (PEASOUP). Our collaboration with this project has yielded several interesting results.
    - PEASOUP deals extensively with concolic testing and smart fuzzing. We have done preliminary investigation for using this technology for Field Certification, but plan to use it more extensively in the next year.
    - We have also developed a program rewriting interface for Strata that we call SPRI. We plan to use this interface to help prevent malicious time bombs, logic bombs, and other malicious codes from becoming active and resulting in system compromise.

Conclusions

Field certification is a paradigm shift in the traditional software development lifecycle and would represent a significant advancement to the state of the art in software assurance. Extending the certification process to include post-deployment provides the ability to safely executed untrusted code in Army critical applications.

Our progress to date includes several mechanisms and observations about characterizing an application, as well as investigating preliminary approaches to assisting the acceptance test engineers in their duties. We have identified several key problems in characterizing applications, including the path explosion problem. Further, we have layed out plans to explore these issues in the upcoming year.

Technology Transfer

Our techniques are predicated on the ability to perform low overhead software dynamic translation as an application runs. This technology, although still emerging, is already in use in industry and has demonstrated its ability to perform reliably and efficiently. Previous work in this area has yielded two patent applications. Consequently, we believe, once completed, that our techniques could be practically applied in a variety of realms with minimal future research effort.

Both PIs have companies (Davidson-Zephyr Software; Knight-Dependable Computing) that are actively engaged in commercialization of these technologies through aggressive pursuit of relevant Small Business Innovative Research (SBIR) funds as well as strategic partnerships with larger companies (e.g., Raytheon, SAIC, Symantec).

Zephyr Software has now received two Phase II SBIR contracts. The first is being award by the Office of Naval Research. The title of the project, "Preventing Program Hijacking via Static and Dynamic Analysis" takes the techniques developed as part of this Army Research Office project and applies it to prevent a very common mode of attack used by malware—subverting control flow of the application to effect malicious actions. Our industrial partner on this project is Raytheon, IDS Cyber Solutions and Integration Division.

In February 2014, Zephyr Software was selected by DARPA for Phase II funding of the project, "Embedded Systems Protection." The focus of this project is to deal with the challenges of protecting embedded systems, which often have major constraints such as memory constraints, real-time constraints, and power constraints that must be considered when protecting them. Our industrial partner is Raytheon, IDS Cyber Solutions and Integration Division.